Shahmeer Ali Akhtar (21i-0466)

Implementation Process:

The implementation involved processing a text dataset by loading multiple files, converting text to lowercase, and removing unnecessary numbers. The text was then tokenized using the NLTK library to prepare it for further analysis. After tokenization, unigram, bigram, and trigram models were created using the ngrams function from NLTK. Each model was constructed by counting occurrences of word sequences in the dataset and then computing probability distributions for each n-gram. Additionally, a backward bigram model was implemented by reversing the order of word pairs to analyze word dependencies in a non-traditional way. The probability distributions for bigrams, trigrams, and backward bigrams were computed to determine the likelihood of word sequences and improve language prediction accuracy.

Challenges Faced:

The first challenge was setting up the NLTK library and ensuring that all necessary dependencies were installed correctly. This included downloading language processing resources required for tokenization. Next, understanding how to properly use NLTK for text tokenization and n-gram extraction required some learning, particularly in handling text normalization and ensuring proper tokenization boundaries. Another significant challenge was implementing and understanding perplexity calculations, which measure how well a model predicts a given test set. Computing perplexity required handling probability distributions correctly and avoiding errors due to zero probabilities in sparse datasets.

An additional challenge was implementing the backward bigram model. Unlike the standard bigram model, which predicts the next word based on the previous word, the backward bigram model predicts the previous word based on the next word. This required carefully modifying the data structure to store reversed word pairs. Generating meaningful sentences using the backward bigram model was also difficult since natural language tends to have a strong forward dependency structure. Ensuring that the reversed model still produced coherent sequences and interpreting its results correctly required additional testing and adjustments.

Comparison of N-Gram Models:

From the results, unigram models are the simplest as they consider only individual words without any contextual relationships. While fast and efficient, unigram models do not capture dependencies between words, making them less effective for generating coherent text. Bigram models perform better by incorporating relationships between consecutive words, improving the predictive power of the model while maintaining relatively low computational cost. Trigram models extend this concept by considering two preceding words, leading to more accurate predictions. However, trigram models require significantly more training data to avoid sparsity issues where certain word combinations may not appear frequently enough for meaningful probability estimation.

The backward bigram model presents an alternative perspective by predicting words in reverse order. While this approach can be useful for specific linguistic analyses, it is generally less effective for generating meaningful text because natural language flows in a forward direction. The perplexity scores obtained in the experiments indicated that higher n-gram models generally performed better in terms of predictive accuracy, but at the cost of increased computational complexity and data dependency. The backward bigram model had limited effectiveness compared to standard bigram and trigram models, highlighting the importance of directional context in language modeling.